FOCS Homework 9, due Day 10

You may edit your answers into this file, or add a separate file in the same directory.

If you add a separate file, please include the following at the top:

```
Student Name: Solution Set
Check one:
[ ] I completed this assignment without assistance or external resources.
[X] I completed this assignment with assistance from ALL COURSE STAFF
   and/or using these external resources: ___
```

1. Reading

Read Sipser pp. 101–125. (This was the optional reading for today. It is required for Monday.)

2. Constructing Grammars

Construct a Context Free Grammar for each of the following languages.

a) All strings (over {0,1}) consisting of a substring *w* followed by its reverse. (This is the same problem you were asked to work on in class.)

Solution:

```
S -> 0S0
S -> 1S1
S -> ϵ
```

Equivalently:

```
S -> 0S0 | 1S1 | ϵ
```

Problem:

Give a derivation for `010010`.

Solution:

```
S -> 0S0 -> 01S10 -> 010ϵ010 = 010010
```

Problem:

b) All strings (over {a,b,c}) of the form a^{i}b^{i}c^{j}: an equal number of `a`s and `b`s, followed by any number of `c`s. For example, `aabb`, `aabbcc`, and `aabbcccc`, but not `aaaabbcc`.

Solution:

G1:

```
S -> TU
T -> aTb | ϵ
U -> cU | ϵ
```

(The final rule could also be `U -> Uc | ϵ`.)

`T` derives any number of `a`s followed by an equal number of `c`s. `U` derives any number of `c`s.

Follow-on question: do *G2* or G3* work? (Are there strings that *G1* derives but *G2* or *G3* do not? Are there strings that *G2* or *G3* derive but *G1* does not?)

G2:

```
S -> ST
S -> aSb | ϵ
T -> cT | ϵ
```

G3:

```
S -> TS
S -> Sc | ϵ
T -> aTb | ϵ
```

Problem:

c) All strings (over {a,b,c}) of the form a^{i}b^{j}c^{j}: any number of `a`s, followed by an equal number of `b`s and `c`s. For example, `abbcc`, `aabbcc`, and `aaaabbcc`, but not `aabbccc`.

Solution:

```
S -> TU
T -> aT | ϵ
U -> bUc | ϵ
```

Problem:

d) Give two distinct grammars that produce the strings described by the regular expression `(ab)*`: empty, `ab`, `abab`, `ababab`, ...

Solution:

Any two of:

G1:

```
S -> abS | ϵ
```

G2:

```
S -> Sab | ϵ
```

G3:

```
S -> T
T -> abT | ϵ
```

G4:

```
S -> abT | ϵ
T -> abT | ϵ
```

G5:

```
S -> abT | ϵ
T -> abS | ϵ
```

G6:

```
S -> aT | ϵ
T -> bS
```

(Would *G5* work if the final rule were `T -> bS | ϵ`?)

G7:

```
S -> AU | ϵ
T -> AU
U -> BT
A -> a
B -> b
```

(*G7* is in Chomsky normal form.)

4. Ambiguous Grammars

Consider the grammar:

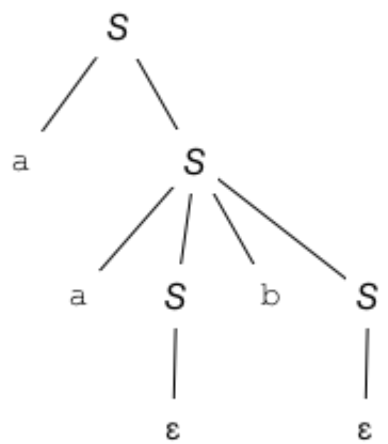
```
S --> a S | a S b S | epsilon
```

This grammar is ambiguous. Show in particular that the string `a a b` has two:

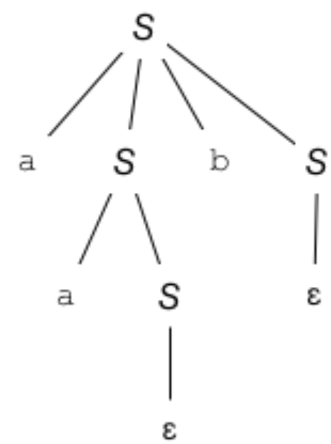
a. parse trees

Solution:

1.



2.



Problem:

b. leftmost derivations (These are the ones that, starting from the start variable, rewrite the leftmost nonterminal first.)

Solution:

1.

```
S -> a S        # using derivation S -> a S
  -> a a S b S  # applying derivation S -> a S b S, to the (only) S
  -> a a   b S  # applying derivation S -> epsilon, to the leftmost S
  -> a a   b    # applying derivation S -> epsilon, to the remaining S
```

2.

```
S -> a S   b S  # using derivation S -> a S b S
  -> a a S b S  # applying derivation S -> a S, to the leftmost S
  -> a a   b S  # applying derivation S -> epsilon, to the leftmost S
  -> a a   b    # applying derivation S -> epsilon, to the remaining S
```

Problem:

c. rightmost derivations (These are the ones that, starting from the start variable, rewrite the rightmost nonterminal first.)

Solution:

1.

```
S -> a S        # using derivation S -> a S
  -> a a S b S  # applying derivation S -> a S b S, to the (only) S
  -> a a S b    # applying derivation S -> epsilon, to the rightmost S
  -> a a   b    # applying derivation S -> epsilon, to the remaining S
```

2.

```
S -> a S   b S  # using derivation S -> a S b S
  -> a S   b    # applying derivation S -> epsilon, to the rightmost S
  -> a a S b    # applying derivation S -> a S, to the rightmost S
  -> a a   b    # applying derivation S -> epsilon, to the rightmost S
```

Extra Credit/Challenge: Prove that this grammar generates all and only the strings of `a`s and `b`s such that every prefix has at least as many `a`s as `b`s. **Hint: Do the readings!**

5. [Optional] Play with Prolog

Read the page and download the sample files from here.

Warning: your instructor was unable to get these to reliably run today, and some of the instructions are from memory of when it previously worked. Your mileage may vary.